Homework 1: Finding Similar Items: Textually Similar Documents

Yage Hao (yage@kth.se)

Hongyi Luo (hongyi@kth.se)

1. A Short Description of the Program

Our program is based on python, using Jupiter notebook as an interface. It is used to find the similarity between two textual documents. Here, we randomly picked 8 text files from 'Opinosis Opinion／Review Data Set' (https://archive.ics.uci.edu/ml/datasets/Opinosis+Opinion+%26frasl%3B+Review) as our dataset in order to test our implementation.

2. Instructions on build and run

2.0. Data Preparation

Our data are reviews (strings) contained in 8 .txt files. In this stage, we import and store them in a python list. This list (dataset) contained 8 elements, each corresponding to one original document.

2.1. Shingling

In this part, we first construct a function '*shingling*', with a document inputted and a shingle size, k, chose. This function will return a list containing all shingles the input document has. For example, if we input text = [similar items] with k=5, the output will be ['simil', 'imila', 'milar', 'ilar ', 'lar I', 'ar it', 'r ite', ' item', 'items'].

Then we apply the above '*shingling*' function to all our 8 documents, receiving a list of all possible shingles. And to ease the computation and storage, we convert the unique shingles into hash values using python default hash function.

2.2. Compare Jaccard Similarity

We construct a function '*jaccard_similarity*'. With two textual documents inputted, it computes Jaccard Similarity between two inputs based on the definition: intersection between two sets divided by union of two sets.

2.3. MinHashing

In this step, we build a minhash signature to replace the previous hash shingle set. The function '*isPrime*' and *'random_hash'* together generate n hash functions in the form of 'ax+b mod c, c is prime'. Then with these n hash functions, we can build signature matrix, where each row corresponding to a hash function and each column corresponding to a document. In this step, we implied MinHash algorithm.

2.4. Compare minhash signatures

To compare minhash signatures, we build a function *'vector_similarity'*. It counts the number of same entry between two columns of signature matrix, and then is divided by n (the number of hash functions we applied in minhash) to give a probability that two inputs are the same, which is an approximation to Jaccard similarity.

2.5. (Optional) LSH

We implement a function *'lsh'* with input argument number of bands. This function reduces computation of similarity between every two columns, instead only compute similarity between candidate pairs, which are columns in a band hashed into the same bucket. If the similarity between two columns with candidate pairs is higher than the threshold, we print them as similar items.

3. Result

3.0. Data Preparation

The original .txt document information:

Label	Document name	Text length (Characters with space)
D1	battery-life_amazon_kindle	9571
D2	battery-life_ipod_nano_8gb	6229
D3	location_bestwestern_hotel_sfo	30013
D4	location_holiday_inn_london	37191
D5	price_amazon_kindle	10484
D6	room_holiday_inn_london	65403
D7	rooms_bestwestern_hotel_sfo	25725
D8	screen_ipod_nano_8gb	5737

3.1. Shingling

In constructing k-shingles, we choose shingle size k=5. Since our document is composed of many short comments and reviews, k=5 will give us a good accuracy.

Label	Document name	Number of 5-Shingles
D1	battery-life_amazon_kindle	4838
D2	battery-life_ipod_nano_8gb	3228
D3	location_bestwestern_hotel_sfo	8507
D4	location_holiday_inn_london	9910
D5	price_amazon_kindle	5360
D6	room_holiday_inn_london	17206
D7	rooms_bestwestern_hotel_sfo	9323
D8	screen_ipod_nano_8gb	3247

Number of total 5-Shingles: 61619. Number of unique 5-Shingles: 30623.

Then the 30623 unique 5-Shingles will be hashed and for the further computation, we will use the hashed shingle sets.

3.2. Compare Jaccard Similarity

Jaccard Similarity Matrix:

```
[[1.    0.173 0.147 0.156 0.199 0.149 0.151 0.141]
 [0.173 1.    0.123 0.125 0.149 0.108 0.122 0.225]
 [0.147 0.123 1.    0.291 0.166 0.231 0.294 0.114]
 [0.156 0.125 0.291 1.    0.174 0.286 0.266 0.122]
 [0.199 0.149 0.166 0.174 1.    0.163 0.172 0.144]
 [0.149 0.108 0.231 0.286 0.163 1.    0.286 0.1  ]
 [0.151 0.122 0.294 0.266 0.172 0.286 1.    0.115]
 [0.141 0.225 0.114 0.122 0.144 0.1   0.115 1.   ]]
```

3.3. MinHashing

We tried several parameter choices and finally pick number of hash functions n=1000 and upper bound of a, b, and c as 100000. This will give us a reasonable result and an acceptable running time on my laptop. The output dimension of signature matrix is 1000 x 8.

3.4. Compare minhash signatures

MinHashing signature similarity matrix:

```
[[1.    0.293 0.307 0.304 0.316 0.296 0.295 0.26 ]
 [0.293 1.    0.24  0.238 0.251 0.239 0.237 0.33 ]
 [0.307 0.24  1.    0.425 0.299 0.41  0.432 0.236]
 [0.304 0.238 0.425 1.    0.33  0.479 0.414 0.237]
 [0.316 0.251 0.299 0.33  1.    0.311 0.305 0.227]
 [0.296 0.239 0.41  0.479 0.311 1.    0.458 0.232]
 [0.295 0.237 0.432 0.414 0.305 0.458 1.    0.226]
 [0.26  0.33  0.236 0.237 0.227 0.232 0.226 1.   ]]
```

3.5. (Optional) LSH

Since in previous MinHash step, we picked number of hash function n=1000. Here, we set number of bands bands=200. Therefore, number of rows in each band is 5, and threshold is about 0.35.

```
Document D6 and D7 are similar. Estimation of similarity through minhash is 0.4
58. Jaccard similarity is 0.2858181465684374.
Document D3 and D7 are similar. Estimation of similarity through minhash is 0.4
32. Jaccard similarity is 0.2938103185545316.
Document D4 and D6 are similar. Estimation of similarity through minhash is 0.4
79. Jaccard similarity is 0.28572783309625416.
Document D4 and D7 are similar. Estimation of similarity through minhash is 0.4
14. Jaccard similarity is 0.26566201632008424.
Document D3 and D6 are similar. Estimation of similarity through minhash is 0.4
1. Jaccard similarity is 0.23146551724137931.
Document D3 and D4 are similar. Estimation of similarity through minhash is 0.4
25. Jaccard similarity is 0.2905192348118562.
```